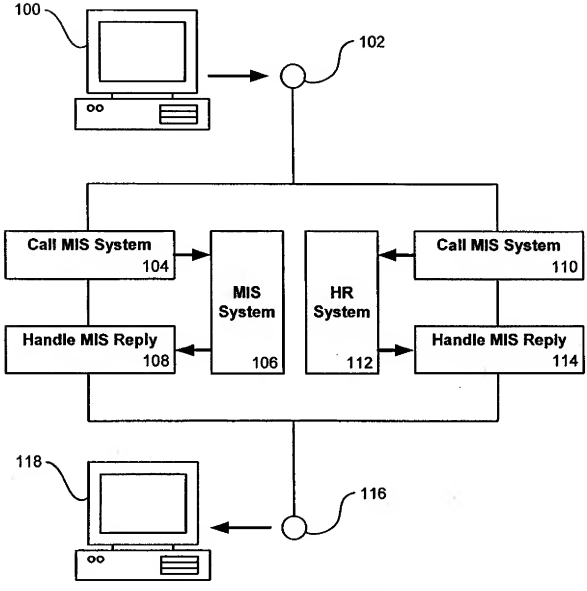
100
102
Call MIS System
104
MIS
System
106
HR
System
112
Call MIS System
110
Handle MIS Reply
108
Handle MIS Reply
114
118
116

**Figure 1**

```
/**
*       @jwf:flow flow::
*
*       <process>
*       <receive name="Receive employee info" method ="start"/>
*       <parallel join-condition="AND">
*               <branch>
*               <perform name="Request e-mail" method="reqMail"/>
*               <receive name="Get e-mail" method="MIS_mailReady"/>
*               <perform name="Notify client about e-mail"
*                       method ="mailNotification"/>
*               </branch>
*               <branch>
*               <perform name="Request benefits" method ="regBenefits"/>
*               <receive name="Get benefits" method ="hr_benefitsReady"/>
*               <perform name="Notify client about benefits"
*                       method ="benefitsNotification"/>
*               </branch>
*       </parallel> * <perform name="Reply to requestor" method="end"/>
*       </process>
* ::
**/
```

**Figure 2**

120

```
*       @jwf:flow flow::
*
*       <process name="PurchaseOrder">
*       <receive name="Receive PO" method="getPO"/>
* *     <forEach name="process Line Items" var="lineitem"
*             expression="getLineItems"
*             parameters="inputPO">
*       <perform name="Process line item" method="processOrder"/>
*             <receive name="Handle service ack."
*                   method="orderService sendAck"/>
*       </forEach>
*       <perform name="Send reply to the client" method="send Reply"/>
*       </process> *
*       ::
*
*       xquery::
*       define function getLineItems (element $po) returns element* {
*             $po/DATAAREA/PROCESS PO/POORDERLIN }
*       define function concat (element $x1, element $x2) returns element {
*             $xl + $x2 }
*       define function buildReply (element $x1) returns element {
*             <reply>$xl <lreply> }
*       ::
**/
```

122

Figure 3B

**Figure 3A**

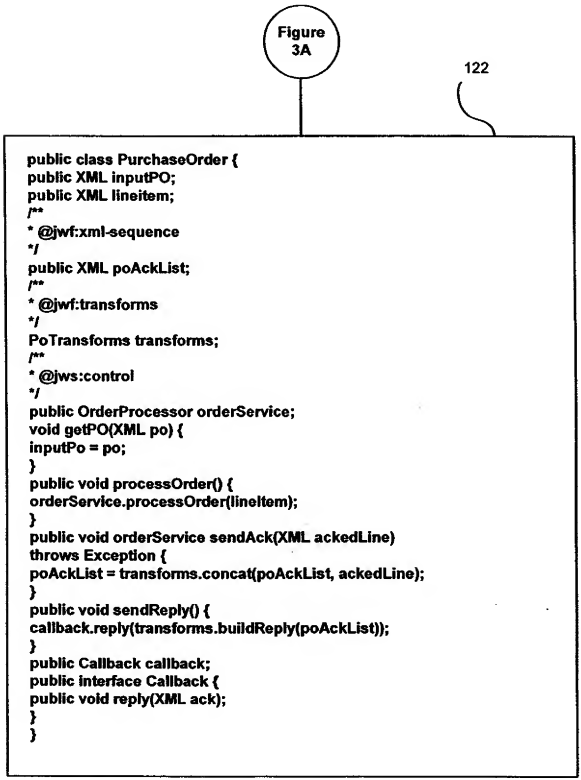
Figure 3A
122
public class PurchaseOrder {
public XML inputPO;
public XML lineitem;
/**
* @jwf:xml-sequence
*/
public XML poAckList;
/**
* @jwf:transforms
*/
PoTransforms transforms;
/**
* @jws:control
*/
public OrderProcessor orderService;
void getPO(XML po) {
inputPo = po;
}
public void processOrder() {
orderService.processOrder(lineItem);
}
public void orderService sendAck(XML ackedLine)
throws Exception {
poAckList = transforms.concat(poAckList, ackedLine);
}
public void sendReply() {
callback.reply(transforms.buildReply(poAckList));
}
public Callback callback;
public interface Callback {
public void reply(XML ack);
}
}

**Figure 3B**